第 1 章 HDFS 概述

1.3 HDFS 组成架构

1）NameNode（nn）：就是Master，它是一个主管、管理者。

（1）管理HDFS的名称空间；

（2）配置副本策略；

（3）管理数据块（Block）映射信息；

（4）处理客户端读写请求。

2）DataNode：就是Slave。NameNode 下达命令，DataNode执行实际的操作。

（1）存储实际的数据块；

（2）执行数据块的读/写操作。

3）Client：就是客户端。

（1）文件切分。文件上传HDFS的时候，Client将文件切分成一个一个的Block，然后进行上传；

（2）与NameNode交互，获取文件的位置信息；

（3）与DataNode交互，读取或者写入数据；

（4）Client提供一些命令来管理HDFS，比如NameNode格式化；

（5）Client可以通过一些命令来访问HDFS，比如对HDFS增删查改操作；

4）Secondary NameNode：并非NameNode的热备。当NameNode挂掉的时候，它并不 能马上替换NameNode并提供服务。

（1）辅助NameNode，分担其工作量，比如定期合并Fsimage和Edits，并推送给NameNode ； （2）在紧急情况下，可辅助恢复NameNode。

1.2 HDFS 文件块大小

HDFS中的文件在物理上是分块存储（Block），块的大小可以通过配置参数 ( dfs.blocksize）来规定，默认大小在Hadoop2.x/3.x版本中是128M，1.x版本中是64M。

第 2 章 HDFS 的 Shell 操作

1）-help：输出这个命令参数 hadoop fs -help rm

2）-moveFromLocal：从本地剪切粘贴到 HDFS

hadoop fs -moveFromLocal ./shuguo.txt /sanguo

3）-copyFromLocal：从本地文件系统中拷贝文件到 HDFS 路径去

hadoop fs -copyFromLocal weiguo.txt /sanguo

4）-put：等同于 copyFromLocal，生产环境更习惯用 put

hadoop fs -put ./wuguo.txt /sanguo

5）-appendToFile：追加一个文件到已经存在的文件末尾

hadoop fs -appendToFile liubei.txt /sanguo/shuguo.txt

6）-copyToLocal：从 HDFS 拷贝到本地

hadoop fs -copyToLocal /sanguo/shuguo.txt ./

7）-get：等同于 copyToLocal，生产环境更习惯用 get

hadoop fs -get /sanguo/shuguo.txt ./shuguo2.txt

8）-ls: 显示目录信息

hadoop fs -ls /sanguo

9）-cat：显示文件内容

hadoop fs -cat /sanguo/shuguo.txt

10）-chgrp、-chmod、-chown：Linux 文件系统中的用法一样，修改文件所属权限

hadoop fs -chmod 666 /sanguo/shuguo.txt

hadoop fs -chown atguigu:atguigu /sanguo/shuguo.txt

11）-mkdir：创建路径

hadoop fs -mkdir /jinguo

12）-cp：从 HDFS 的一个路径拷贝到 HDFS 的另一个路径

hadoop fs -cp /sanguo/shuguo.txt /jinguo

13）-mv：在 HDFS 目录中移动文件

hadoop fs -mv /sanguo/wuguo.txt /jinguo

14）-tail：显示一个文件的末尾 1kb 的数据

hadoop fs -tail /jinguo/shuguo.txt

15）-rm：删除文件或文件夹

hadoop fs -rm /sanguo/shuguo.txt

16）-rm -r：递归删除目录及目录里面内容

hadoop fs -rm -r /sanguo

17）-du 统计文件夹的大小信息

hadoop fs -du -s -h /jinguo --------总文件

hadoop fs -du -h /jinguo ---------每一个文件

18）-setrep：设置 HDFS 中文件的副本数量

hadoop fs -setrep 10 /jinguo/shuguo.txt

这里设置的副本数只是记录在 NameNode 的元数据中，是否真的会有这么多副本，还得 看 DataNode 的数量。因为目前只有 3 台设备，最多也就 3 个副本，只有节点数的增加到 10 台时，副本数才能达到 10。

第 3 章 HDFS 的 API 操作

1）拷贝 hadoop-3.1.0 到非中文路径（比如 d:\）。

2）配置 HADOOP\_HOME 环境变量

3）配置 Path 环境变量。 注意：如果环境变量不起作用，可以重启电脑试试。

验证 Hadoop 环境变量是否正常。双击 winutils.exe，如果报如下错误。说明缺少微软运行库（正版系统往往有这个问题）。再资料包里面有对应的微软运行库安装包双击安装即可。

4）在 IDEA 中创建一个 Maven 工程，并导入相应的依赖坐标+日志添加

<dependencies>

<dependency>

<groupId>org.apache.hadoop</groupId>

<artifactId>hadoop-client</artifactId>

<version>3.1.3</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

<version>1.7.30</version>

</dependency>

</dependencies>

在项目的 src/main/resources 目录下，新建一个文件，命名为“log4j.properties”

log4j.rootLogger=INFO, stdout

log4j.appender.stdout=org.apache.log4j.ConsoleAppender

log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern=%d %p [%c] - %m%n

log4j.appender.logfile=org.apache.log4j.FileAppender

log4j.appender.logfile.File=target/spring.log

log4j.appender.logfile.layout=org.apache.log4j.PatternLayout

log4j.appender.logfile.layout.ConversionPattern=%d %p [%c] - %m%n

客户端去操作 HDFS 时，是有一个用户身份的。默认情况下，HDFS 客户端 API 会从采 用 Windows 默认用户访问 HDFS，会报权限异常错误。所以在访问 HDFS 时，一定要配置 用户。

// 1 获取文件系统

Configuration configuration = new Configuration();

// FileSystem fs = FileSystem.get(new URI("hdfs://hadoop102:8020"), configuration);

FileSystem fs = FileSystem.get(new URI("hdfs://hadoop102:8020"), configuration,"silence");

// 创建目录

fs.mkdirs(new Path("/xiyou/huaguoshan/"));

// 执行下载操作

// boolean delSrc 指是否将原文件删除

// Path src 指要下载的文件路径

// Path dst 指将文件下载到的路径

// boolean useRawLocalFileSystem 是否开启文件校验

fs.copyToLocalFile(false, new Path("/xiyou/huaguoshan/sunwukong.txt"), new Path("d:/sunwukong2.txt"), true);

// 执行删除

fs.delete(new Path("/xiyou"), true);

// 获取文件详情

RemoteIterator listFiles = fs.listFiles(new Path("/"), true);

while (listFiles.hasNext())

{ LocatedFileStatus fileStatus = listFiles.next();

System.out.println("========" + fileStatus.getPath() + "=========");

System.out.println(fileStatus.getPermission());

System.out.println(fileStatus.getOwner());

System.out.println(fileStatus.getGroup());

System.out.println(fileStatus.getLen());

System.out.println(fileStatus.getModificationTime());

System.out.println(fileStatus.getReplication());

System.out.println(fileStatus.getBlockSize());

System.out.println(fileStatus.getPath().getName());

// 获取块信息

BlockLocation[] blockLocations = fileStatus.getBlockLocations();

System.out.println(Arrays.toString(blockLocations));

} //

// 判断是文件还是文件夹

FileStatus[] listStatus = fs.listStatus(new Path("/"));

for (FileStatus fileStatus : listStatus)

{ // 如果是文件

if (fileStatus.isFile())

{ System.out.println("f:"+fileStatus.getPath().getName()); }

else { System.out.println("d:"+fileStatus.getPath().getName()); } }

// 关闭资源

fs.close();

5）参数优先级 参数优先级排序：

（1）客户端代码中设置的值 >（2）ClassPath 下的用户自定义配置文 件 >（3）然后是服务器的自定义配置（xxx-site.xml）>（4）服务器的默认配置（xxx-default.xml）

第 4 章 HDFS 的读写流程

4.1 HDFS 写数据流程

（1）客户端通过 Distributed FileSystem 模块向 NameNode 请求上传文件，NameNode 检查目标文件是否已存在，父目录是否存在。

（2）NameNode 返回是否可以上传。

（3）客户端请求第一个 Block 上传到哪几个 DataNode 服务器上。

（4）NameNode 返回 3 个 DataNode 节点，分别为 dn1、dn2、dn3。

（5）客户端通过 FSDataOutputStream 模块请求 dn1 上传数据，dn1 收到请求会继续调用 dn2，然后 dn2 调用 dn3，将这个通信管道建立完成。

（6）dn1、dn2、dn3 逐级应答客户端。

（7）客户端开始往 dn1 上传第一个 Block（先从磁盘读取数据放到一个本地内存缓存）， 以 Packet 为单位，dn1 收到一个 Packet 就会传给 dn2，dn2 传给 dn3；dn1 每传一个 packet 会放入一个应答队列等待应答。

（8）当一个 Block 传输完成之后，客户端再次请求 NameNode 上传第二个 Block 的服务 器。（重复执行 3-7 步）

4.2 HDFS 读数据流程

（1）客户端通过 DistributedFileSystem 向 NameNode 请求下载文件，NameNode 通过查 询元数据，找到文件块所在的 DataNode 地址。

（2）挑选一台 DataNode（就近原则，然后随机）服务器，请求读取数据

（3）DataNode 开始传输数据给客户端（从磁盘里面读取数据输入流，以 Packet 为单位 来做校验）。

（4）客户端以 Packet 为单位接收，先在本地缓存，然后写入目标文件。

第 5 章 NameNode 和 SecondaryNameNode

（1）基本语法 hdfs oiv -p 文件类型 -i 镜像文件 -o 转换后文件输出路径

hdfs oiv -p XML -i fsimage\_0000000000000000025 -o /opt/module/hadoop-3.1.3/fsimage.xml

（2）基本语法 hdfs oev -p 文件类型 -i 编辑日志 -o 转换后文件输出路径

hdfs oev -p XML -i edits\_0000000000000000012-0000000000000000013 -o /opt/module/hadoop3.1.3/edits.xml

第 6 章 DataNode

6.1 DataNode 工作机制

（1）一个数据块在 DataNode 上以文件形式存储在磁盘上，包括两个文件，一个是数据 本身，一个是元数据包括数据块的长度，块数据的校验和，以及时间戳

（2）DataNode 启动后向 NameNode 注册，通过后，周期性（6 小时）的向 NameNode 上 报所有的块信息。

DN 向 NN 汇报当前解读信息的时间间隔，默认 6 小时；

<property>

<name>dfs.blockreport.intervalMsec</name>

<value>21600000</value>

<description>Determines block reporting interval in

milliseconds.</description>

</property>

DN 扫描自己节点块信息列表的时间，默认 6 小时

<property>

<name>dfs.datanode.directoryscan.interval</name>

<value>21600s</value>

<description>Interval in seconds for Datanode to scan data

directories and reconcile the difference between blocks in memory and on

the disk.

Support multiple time unit suffix(case insensitive), as described

in dfs.heartbeat.interval.

</description>

</property>

（3）心跳是每 3 秒一次，心跳返回结果带有 NameNode 给该 DataNode 的命令如复制块 数据到另一台机器，或删除某个数据块。如果超过 10 分钟没有收到某个 DataNode 的心跳， 则认为该节点不可用。

（4）集群运行中可以安全加入和退出一些机器

6.2 数据完整性

常见的校验算法 crc（32），md5（128），sha1（160）

6.3 掉线时限参数设置

TimeOut = 2 \* dfs.namenode.heartbeat.recheck-interval + 10 \* dfs.heartbeat.interval。

而默认的dfs.namenode.heartbeat.recheck-interval 大小为5分钟，dfs.heartbeat.interval默认为3秒。

需要注意的是 hdfs-site.xml 配置文件中的 heartbeat.recheck.interval 的单位为毫秒， dfs.heartbeat.interval 的单位为秒。

<property>

<name>dfs.namenode.heartbeat.recheck-interval</name>

<value>300000</value>

</property>

<property>

<name>dfs.heartbeat.interval</name>

<value>3</value>

</property>